?

Creating Maps with PHP MapScript

There are few people who don't find maps intriguing, especially old maps with a antique look to them. So it is quite natural that digital mapping has become so popular. Where would we be without visualizing election results on a map? How effective would it be to hear about current affairs around the world without seeing how the countries involved are geographically related to their neighbors? The Google Maps and Mapquest.com's of the world are a popular incarnation of some exciting web mapping technology that you too can use. With a few pieces of mapping data, a mapping programming library and a PHP script, you can create custom and interactive maps.

Open Source Mapping Tools

There are a handful of open source mapping tools available, from desktop applications to web-enabled mapping services. One popular web mapping application is the UMN MapServer (http://ms.gis.umn.edu). With a very large user base, an active community and dedicated developers, it is a powerful product for publishing maps over the web.

PHP-based web mapping tools

MapServer is actively used as the backend to many PHP web page front-ends. For example, the Chameleon (http://maptools.org) and MapBender (http://mapbender.org) products both use PHP extensively. There is also a powerful implementation of AJAX-based web mapping called ka-Map available. (See my tutorial on using ka-Map at: http://www.xml.com/pub/a/2005/08/10/ka-map.html). They allow us to handle maps and mapping data like we would other web-based applications, by wrapping it all up with PHP. MapServer still sits in the back-end, happily cranking out map images, while PHP controls interaction and brokers requests.

MapScript: MapServer's API

MapServer is commonly used as a CGI application coordinating with a web server. To have ultimate powerful and flexibility you can use MapServer API with one of many programming languages, including PHP, Perl, Python, Ruby, Java, C# and more. MapScript provides methods for interacting with mapping data, cartographic styling of map output and creation of final map images.

MapServer concepts

MapServer's core configuration is through a text-based runtime configuration file. Referred to as a map file, it is the core of most MapServer-based applications. The CGI program or custom MapScript application reads this configuration information, accesses data, draws the map and returns a graphic ready for online viewing. This can also be run as a stand-alone command line program without any connection to the web. The examples in this hack use PHP MapScript from the command line. You can take these examples and modify them to suit your custom web environment.

The map file has a simple hierarchical object structure, with object inheriting settings from their children. This hack uses a very simple map file that is only intended for use with PHP MapScript. To run it in a CGI environment you would need set more settings. You will also learn how to make an application without having a map file, creating it in runtime within a PHP script.

Getting the PHP MapScript extension

Before you can start using PHP MapScript, you need to download and set up MapScript tools. The main PHP MapScript site has download instructions as well as an API reference document: http://maptools.org/php_mapscript/. There are at least two other easy ways to get binary distributions of PHP and MapScript, pre-configured to work together.

For Windows, you can use the MapServer for Windows (MS4W) distribution: http://ms4w.maptools.org. It comes as a base zip file containing all the MapServer basics, as well as PHP MapScript.

For Linux, you can use the FGS Linux installer at: http://maptools.org/fgs. It includes an install shell script that you can use to help automate the install. FGS runs as a separate set of libraries, applications and even a web server, making it easy to get started without having to compile a bunch of external dependencies.

For other operating systems you may need to compile your own MapServer from source. This will, optionally, create the PHP MapScript libraries and is not always simple to do.

Hacking maps with PHP

The following hacks show how to use map files, manipulate map files and create maps from scratch all within a simple PHP script.

Step 1: Data prep

Before you can start creating a map you will need some base map information. For these hacks, only one file is required. This is an image created from global satellite imagery and comes in the form of a GeoTIFF file. Figure 1 shows an example of the global cloud image dataset.

Figure 1. The global cloud image dataset.

The TIFF image format may already be familiar to you, but GeoTIFF extends it further, allowing the image to have geographic coordinate information embedded in the image. This makes it possible to use as mapping data file where you might want to look at a particular geographic area (say, a certain latitude and longitude) - GeoTIFF's can quickly make that translation between pixel rows and columns and geographic X and Y coordinates.

You can download the sample cloud image dataset from: http://spatialguru.com/maps/data/world/weather/cloud_image.zip

CAN WE PUT THIS DATA FILE SOMEWHERE ON OREILLYNET INSTEAD?

For other data sources including satellite imagery, country boundary lines, remote web services, etc. see my book Web Mapping Illustrated (http://oreilly.com/catalog/webmapping) or online sites like freegis.org.

For these examples, you will put your scripts in one folder and your data in a sub-folder. Create a folder for your scripts and a sub-folder called data to store your image file. Unzip the `cloud_image.zip` file into the data folder, it should create two files. A `.tif` file holding the image and a `.tfw` file that holds some geographic referencing information.

Step 2: Just render a map

A simple MapServer map file and a few bits of PHP code can make you a map. You create a map object in your script which gives you access to several methods for working

with a specific map file. Example 1 shows the listing of the bare minimum PHP code required.

Example 1. Basic PHP script access to a map file for rendering a map image.

```
<?PHP
// ex1_map_basic.php
// Tyler Mitchell, August 2005
// Build a map using a pre-made map file

// Load MapScript extension
if (!extension_loaded("MapScript"))
  dl('php_mapscript.'.PHP_SHLIB_SUFFIX);

// Create a map object.
$oMap = ms_newMapObj("map_points.map");

// Render the map into an image object
$oMapImage = $oMap->draw();

// Save the map to an image file
$oMapImage->saveImage("worldmap.png");

?>
```

As you can see it is really only four lines of PHP code to go through the basic mapping steps: load the MapScript extension, open the map file, render the map image, and save the image to a file. The result of the script is a single image, `worldmap.png`. Open the image in your favorite image viewer and you should see a global image.

All the map settings are done in the map configuration file. Example 2 shows the contents of the map file. It includes a single layer of mapping data using the global map GeoTIFF image. This is an extremely simple example that probably won't work outside of the MapScript environment (i.e. using CGI MapServer). For more details on the options and settings in a MapServer configuration file see the map file reference documentation on the MapServer website: http://ms.gis.umn.edu.

Example 2. Contents of the basic map file.

```
MAP
  NAME MAP_POINTS
  SIZE 600 300
  EXTENT -180 -90 180 90
  LAYER
    NAME clouds
    TYPE RASTER
    STATUS DEFAULT
    DATA "data/global_clouds.tif"
  END
END
```

Step 3: Modify the map

Of course, the real power of mapping capabilities within a programming language like PHP is having the ability to control things. Example 1 didn't change any settings or add anything to what was already in the map file. The map file is static, but with PHP you can add new objects and change ones already defined in the map file.

For example, you might want to change the geographic extent that the map covers. Instead of using the default `EXTENT` setting in the map file, you can create your own. All you need to do is set the map object's extent property to use your custom coordinates. The extent setting requires two pairs of coordinates, the first pair representing the south-west most corner of the map and the second pair the north-east. You add just one line to the script, as shown in Example 3.

Example 3. Changing the default map extent.

```
<?PHP
// ex3_map_change.php
// Tyler Mitchell, August 2005
// Build a map using a pre-made map file
// and change one property

// Load MapScript extension
if (!extension_loaded("MapScript"))
  dl('php_mapscript.'.PHP_SHLIB_SUFFIX);

// Create a map object
$oMap = ms_newMapObj("ex2_map_basic.map");

// Change the map object's extent property
$oMap->setExtent(-130,20, -70,70);

// Render the map into an image object
$oMapImage = $oMap->draw();

// Save the map to an image file
$oMapImage->saveImage("worldmap.png");

?>
```

The resulting map image is shown in Figure 2.

fig2_zoom.png

Figure 2. Map centered on North American extents.

Your script can do much more if you make it interactive for the user. By linking MapScript code into your web applications, you can provide limitless opportunities for zooming in/out of the map, turning layers on and off, adding more data, changing colours, etc. Refer to the PHP MapScript documentation to get an idea of other methods and properties.

Step 4: Mapping from scratch

You can also create a full PHP MapScript application from scratch without using an existing map file. This can serve two purposes. You can use the script to create a new map file. It is also a good way of keeping your application bundled together without having external map files to deal with. As you might guess, you will have to create all the objects and set all the properties using PHP code, which will be slightly larger than if you simply used a map file, but provides you instant control over all the settings, right in the PHP code.

One common use of PHP MapScript applications is for generating dynamic point locations on a map. There are methods for creating what are called *inline features*, where the coordinates of a location (points, lines and polygon areas are all possible) are entered into the map file. Example 4 shows a map file where a single point and an accompanying text label are provided. The map file also includes a symbol object, this will tell MapScript how to paint the points on the map. The real work is done in the new layer object at the end.

Example 4. A map file with an additional layer showing an inline feature.

```
MAP
  NAME MAP_POINTS
  SIZE 600 300
  EXTENT -180 -90 180 90

  SYMBOL
    NAME "circle"
    TYPE ELLIPSE
    FILLED TRUE
    POINTS
      1 1
    END
  END

  LAYER
    NAME clouds
    TYPE RASTER
    STATUS DEFAULT
    DATA "data/global_clouds.tif"
  END

  LAYER
    NAME custom_points
    TYPE POINT
    STATUS DEFAULT
    FEATURE # Inline feature definition
      POINTS
        -121 54
      END
      TEXT "My Place"
    END
```

```
    CLASS
      COLOR 250 0 0
      OUTLINECOLOR 255 255 255
      SYMBOL "circle"
      SIZE 10
      LABEL
        POSITION AUTO
        COLOR 250 0 0
        OUTLINECOLOR 255 255 255
      END
    END
  END
END
```

The resulting map is shown in Figure 3.

fig3_inline_point.png

Figure 3. Map showing the inline point and label that was added to the map.

Don't be too distracted by the symbol setting. While it looks obscure, you will likely not need to create additional ones until you start doing more advanced line symbology or area shading.

While handy, a map file is not inherently dynamic, this is where PHP comes in. Example 5 is an extensive example showing how to create the entire map all using PHP and no map file. It also includes a section that will pull coordinates from a text file and dynamically add them to the map when it is rendered. Example 6 shows the contents of the simple text file (`points.txt`) containing the longitude/latitude point coordinates and a text label, delimited with commas (and with no header line). This file should be saved in the data sub-folder.

Example 5. PHP script showing how to set a number of map settings without using a map file and importing points from a text file.

```
<?PHP
// ex5_map_points.php
// Build a map using a single GeoTIFF
// and a text file of coordinates/labels.
// Does not require a mapserver map file to run.
// Tyler Mitchell, August, 2005

// Load MapScript extension
if (!extension_loaded("MapScript"))
  dl('php_mapscript.'.PHP_SHLIB_SUFFIX);

// Create a map object. Provide empty string if not
// using an existing map file
$oMap = ms_newMapObj("");

// Set size of the output map image
$oMap->setSize(600,300);

// Set the geographic extents of the map.
```

```
$oMap->setExtent(-180,-90,180,90);

// Create a map symbol, used as a brush pattern
// for drawing map features (lines, points, etc.)
$nSymbolId = ms_newSymbolObj($oMap, "circle");
$oSymbol = $oMap->getsymbolobjectbyid($nSymbolId);
$oSymbol->set("type", MS_SYMBOL_ELLIPSE);
$oSymbol->set("filled", MS_TRUE);
$aPoints[0] = 1;
$aPoints[1] = 1;
$oSymbol->setpoints($aPoints);

// Create a data layer and associate it with the map.
// This is the raster layer showing some cloud imagery
$oLayerClouds = ms_newLayerObj($oMap);
$oLayerClouds->set( "name", "clouds");
$oLayerClouds->set( "type", MS_LAYER_RASTER);
$oLayerClouds->set( "status", MS_DEFAULT);
$oLayerClouds->set( "data","data/global_clouds.tif");

// Create another layer to hold point locations
$oLayerPoints = ms_newLayerObj($oMap);
$oLayerPoints->set( "name", "custom_points");
$oLayerPoints->set( "type", MS_LAYER_POINT);
$oLayerPoints->set( "status", MS_DEFAULT);

// Open file with coordinates and label text (x,y,label)
$fPointList = file("data/points.txt");

// For each line in the text file
foreach ($fPointList as $sPointItem)
{
   $aPointArray = explode(",",$sPointItem);
   // :TRICKY: Although we are creating points
   // we are required to use a line object (newLineObj)
   // with only one point. I call it a CoordList object
   // for simplicity since we aren't really drawing a line.
   $oCoordList = ms_newLineObj();
   $oPointShape = ms_newShapeObj(MS_SHAPE_POINT);
   $oCoordList->addXY($aPointArray[0],$aPointArray[1]);
   $oPointShape->add($oCoordList);
   $oPointShape->set( "text", chop($aPointArray[2]));
   $oLayerPoints->addFeature($oPointShape);
}

// Create a class object to set feature drawing styles.
$oMapClass = ms_newClassObj($oLayerPoints);

// Create a style object defining how to draw features
$oPointStyle = ms_newStyleObj($oMapClass);
$oPointStyle->color->setRGB(250,0,0);
$oPointStyle->outlinecolor->setRGB(255,255,255);
$oPointStyle->set( "symbolname", "circle");
$oPointStyle->set( "size", "10");
```

```
// Create label settings for drawing text labels
$oMapClass->label->set( "position", MS_AUTO);
$oMapClass->label->color->setRGB(250,0,0);
$oMapClass->label->outlinecolor->setRGB(255,255,255);


// Render the map into an image object
$oMapImage = $oMap->draw();

// Save the map to an image file
$oMapImage->saveImage("worldmap.png");

?>
```

Should remove comments from above code to shorten it up?

Example 6. Listing of the points.txt file.

```
-118.35,34.06,Angie
-118.40,34.03,Ray
-111.99,33.52,Alice
-95.45,29.75,David
144.85,-37.85,Mark
```

The output of the final map is shown in Figure 4. To take this example a step further, you could set up the dynamic point generator as a separate class. Then you could call this class from other PHP scripts, e.g. scripts that take coordinates as user input, and have them drawn on your map.

Figure 4. Map showing points and labels taken from text file.

Learning more

To learn more about MapServer and other open source geospatial technologies, there are many great places to get started. The community actively uses mailing lists, IRC discussion channels and holds annual conferences. To find other users in your area, ask on a mailing list. The MapServer mailing list tends to be a focal point for many other projects as they tend to be intertwined, so it may be the best place to look for help from a human.

Web Mapping Illustrated (O'Reilly, 2005) covers a wide range of information including MapServer, spatial databases, OGC web services, data conversion, map projections and much more. Other books on related (open source) subjects include: *Mapping Hacks* (O'Reilly, 2005), *Beginning MapServer* (Apress, not yet released) and *Pragmatic GIS* (Pragmatic, not yet released).